Asymptotic Quasi-Polynomial Time Approximation Scheme for Resource Minimization for Fire Containment

Mirmahdi Rahgoshay
Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G 2E8, Canada
rahgosha@ualberta.ca

Mohammad R. Salavatipour
Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G 2E8, Canada
mrs@ualberta.ca

Abstract

Resource Minimization Fire Containment (RMFC) is a natural model for optimal inhibition of harmful spreading phenomena on a graph. In the RMFC problem on trees, we are given an undirected tree G, and a vertex r where the fire starts at, called root. At each time step, the firefighters can protect up to B vertices of the graph while the fire spreads from burning vertices to all their neighbors that have not been protected so far. The task is to find the smallest B that allows for saving all the leaves of the tree. The problem is hard to approximate up to any factor better than 2 even on trees unless P = NP [11].

Chalermsook and Chuzhoy [6] presented a Linear Programming based $O(\log^* n)$ approximation for RMFC on trees that matches the integrality gap of the natural Linear Programming relaxation. This was recently improved by Adjiashvili, Baggio, and Zenklusen [1] to a 12-approximation through a combination of LP rounding along with several new techniques.

In this paper we present an asymptotic QPTAS for RMFC on trees. More specifically, let $\epsilon > 0$, and $\mathcal{I}$ be an instance of RMFC where the optimum number of firefighters to save all the leaves is $OPT(\mathcal{I})$. We present an algorithm which uses at most $\lceil (1+\epsilon)OPT(\mathcal{I}) \rceil$ many firefighters at each time step and runs in time $n^{O(\log\log n/\epsilon)}$. This suggests that the existence of an asymptotic PTAS is plausible especially since the exponent is $O(\log\log n)$, not $O(\log n)$.

Our result combines a more powerful height reduction lemma than the one in [1] with LP rounding and dynamic programming to find the solution. We also apply our height reduction lemma to the algorithm provided in [1] plus a more careful analysis to improve their 12-approximation and provide a polynomial time $(5+\epsilon)$-approximation.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Firefighter Problem, Resource Management, Fire Containment, Approximation Algorithm, Asymptotic Approximation Scheme

Digital Object Identifier 10.4230/LIPIcs.STACS.2020.33

Funding Authors supported by organization NSERC.

1 Introduction

The Firefighter problem and a closely related problem named Resource Minimization Fire Containment (RMFC) are natural models for optimal inhibition of harmful spreading phenomena on a graph. The firefighter problem was formally introduced by Hartnell [9] and later Chalermsook and Chuzhoy [6] defined the RMFC problem. Since then, both problems have received a lot of attention in several research papers, even when the underlying graph is a spanning tree, which is one of the most-studied graph structures in this context and also the focus of this paper.

37th International Symposium on Theoretical Aspects of Computer Science (STACS 2020).
Editors: Christophe Paul and Markus Bläser; Article No. 33; pp. 33:1–33:14


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In both problems (when restricted to trees) we are given a graph $G = (V, E)$, which is a spanning tree, and a vertex $r \in V$, called root. The problem is defined over discretized time steps. At time 0, a fire starts at r and spreads step by step to neighboring vertices. During each time step $1, 2, \ldots$ any non-burning vertex u can be protected, preventing u from burning in any future time step.

In the RMFC problem the task is to determine the smallest number $B \in \mathbb{Z}_{\geq 1}$ such that there is a protection strategy which protects B vertices at each time step while saving all the leaves from catching fire. In this context, B is referred to as the number of firefighters (or budget at each step). In the firefighters problem, given a fixed number of firefighters (i.e. number of vertices that can be protected at each time step) the goal is to find a strategy to maximize the number of vertices saved from catching the fire.

For RMFC on trees, King and MacGillivray [11] showed that it is NP-hard to decide whether one firefighter is sufficient or not. This means that there is no (efficient) approximation algorithm with an approximation factor strictly better than 2, unless P=NP. On the positive side, Chalermsook and Chuzhoy [6] presented an $O(\log^* n)$-approximation, where n is the number of vertices. Their algorithm is based on a natural Linear Programming (LP) relaxation, which is a straightforward adaptation of the one previously used for the Firefighter problem on trees and essentially matches the integrality gap of the underlying LP (the integrality gap of the underlying LP is $\Theta(\log^* n)$ [6]). Recently, Adjiashvili et al. [1] presented a 12-approximation for RMFC, which is the first constant factor approximation for the problem. Their result is obtained through a combination of the known LPs with several new techniques, which allows for efficiently enumerating subsets of super-constant size of a good solution to obtain stronger LPs. They also present a $PTAS$ for the firefighter problem.

1.1 Our Results

In this paper our main focus is on RMFC problem. By using Linear Programming and dynamic programming techniques, we show how to approximate RMFC with a small additive error by presenting a quasi-polynomial time asymptotic approximation scheme (AQPTAS) for it. More specifically our main result is the following theorem:

▶ **Theorem 1.** *For RMFC on trees and for any $\epsilon > 0$ there is an algorithm that finds a solution using $\lceil (1 + O(\epsilon))B \rceil$ firefighters with running time $n^{O(\log \log n/\epsilon)}$, where B is the optimal number of firefighters.*

We will also show how applying our more powerful height reduction lemma to the algorithm used by Adjiashvili et al. [1], plus a more careful analysis, leads to a better constant factor. In particular, we obtain the following:

▶ **Theorem 2.** *For any $\epsilon > 0$, there is a polynomial time $(5 + \epsilon)$-approximation for the RMFC problem on trees.*

Recall that the RMFC problem on trees does not admit better than 2-approximation unless P = NP [11]. However, this does not rule out the possibility of a +1 approximation or an asymptotic PTAS. Our result is an indication that it is plausible that an asymptotic PTAS exists, especially since the exponent is $O(\log \log n)$, not $O(\log n)$ as we don't know any natural problem that admits $n^{O(\log \log n)}$ algorithm but not polynomial time.

We start by introducing a more powerful height reduction transformation than the one used in [1] that allows for transforming the RMFC problem into a more compact and better structured form, by only losing a $(1 + \epsilon)$ factor in terms of approximability. This transformation allows us to identify small substructures, over which we can optimize efficiently,

and having an optimal solution to these subproblems we can define a residual LP with small integrality gap. Then we will show how to apply dynamic programming on the transformed instance to obtain a strategy to protect the nodes at each step to successfully contain the fire and save all the leaves with using only $O(\epsilon B)$ more firefighters at each step. We will apply our more powerful height reduction lemma to the previous combinatorial approach [1] to reach a better constant factor approximation in polynomial time, which is presented in Theorem 2.

1.2 Further Related Work

The Firefighter problem and RMFC, both restricted to trees, are known to be computationally hard problems. More precisely, Finbow, King, MacGillivray and Rizzi [7] showed the NP-hardness for the Firefighter problem on trees even when the maximum degree is three. For RMFC on trees, it is NP-hard to decide whether one firefighter is sufficient or not [11], which implies that the problem is hard to approximate to a factor better than 2.

Several approximation algorithms have been proposed for both of these problems. Hartnell and Li [8] proved that a natural greedy algorithm is a $\frac{1}{2}$-approximation for the Firefighter problem. Later, Cai, Verbin and Yang [3] improved this result to $1 - \frac{1}{e}$, using a natural LP relaxation and dependent randomized rounding. Then Anshelevich, Chakrabarty, Hate and Swamy [2] showed that the Firefighter problem on trees can be interpreted as a monotone submodular function maximization (SFM) problem subject to a partition matroid constraint. This observation yields another $(1-\frac{1}{e})$-approximation by using a recent $(1-\frac{1}{e})$-approximation for monotone SFM subject to a matroid constraint [4, 13].

Chalermsook and Vaz [5] showed that, for any $\epsilon > 0$, the canonical LP used for the Firefighter problem on trees has an integrality gap of $1 - \frac{1}{e} + \epsilon$. This generalized a previous result by Cai, Verbin and Yang [3]. When restricted to some tree topologies this factor $1 - \frac{1}{e}$ was later improved (see [10]) but, for arbitrary trees, that was the best known approximation factor for a few years.

Recently, Adjiashvili, Baggio and Zenklusen [1] have filled the gap between previous approximation ratios and hardness results for the Firefighter problem. In particular, they present approximation ratios that nearly match the hardness results, thus showing that the Firefighter problem can be approximated to factors that are substantially better than the integrality gap of the natural LP. Their results are based on several new techniques, which may be of independent interest.

Assuming a variant of the Unique Games Conjecture (UGC), the RMFC problem in general graphs is hard to approximate within any constant factor, according to a recent work by Lee [12] which is based on a general method of converting an integrality gap instance to a length-control dictatorship test for variants of the s-t cut problem. For further results and related work we refer the reader to [1].

1.3 Organization of the Paper

In Section 2 we start by introducing some preliminaries including a (now standard) Linear Programming relaxation for the problem and then will provide a height reduction lemma. Section 3 will cover our main algorithm to obtain the asymptotic QPTAS. In Appendix A we will show how to apply our height reduction lemma to the previous combinatorial approach of [1] to improve their 12-approximation and provide a $(5 + \epsilon)$-approximation.

2 Preliminaries and Overview of the Algorithm

Recall that we are given a tree $G = (V, E)$ rooted at a vertex r, from which we assume the fire starts. We denote by $\Gamma \subseteq V$ the set of all leaves of the tree. Given an instance $\mathcal{I}$ for RMFC and an integer parameter $B \geq 1$, called the budget or the number of firefighters, at each time step we can "protect" up to B non-burning vertices. Such vertices are protected indefinitely. Our goal is to find the smallest B and a protection strategy such that all the leaves Γ are saved from catching the fire. Observe that we say a vertex u is protected, if we directly place a firefighter in u, and a vertex v is saved when the fire does not reach to u, because of protecting some u on the unique v-r path. This smallest value of B is denoted by $OPT(\mathcal{I})$.

Let $L \in \mathbb{Z}_{\geq 1}$ be the depth of the tree, i.e. the largest distance, in terms of the number of edges, between r and any other vertex in G. After at most L time steps, the fire spreading process will halt. For $\ell \in [L] := \{1, \ldots, L\}$, let $V_\ell \subseteq V$ be the set of all vertices of distance ℓ from r, which we call the ℓ-th level of the instance. We also use $V_{\leq \ell} = \cup_{k=1}^{\ell} V_k$, and we define $V_{\geq \ell}$, $V_{<\ell}$, and $V_{>\ell}$ in the same way. Moreover, for each $1 \leq \ell < L$ and each $u \in V_\ell$, $P_u \subseteq V_{\leq \ell} \setminus \{r\}$ denotes the set of all vertices on the unique u-r path except for the root r, and $T_u \subseteq V_{>\ell}$ denotes the subtree rooted at u, i.e. descendants of u.

2.1 Linear Programming Relaxation

We use the following (standard) Linear Programming (LP) relaxation for the problem that is used in both [6] and [1].

$$\begin{aligned} \min \quad & B & & & (1) \\ & x(P_u) & \geq & 1 \qquad \forall u \in \Gamma \\ & x(V_{\leq \ell}) & \leq & B \cdot \ell \qquad \forall \ell \in [L] \\ & x & \in & \mathbb{R}_{\geq 0}^{V \setminus \{r\}} \end{aligned}$$

Here $x(U) := \sum_{u \in U} x(u)$ for any $U \subseteq V \setminus \{r\}$. Note that with $x \in \{0,1\}^{V \setminus \{r\}}$ and $B \in \mathbb{Z}_{\geq 0}$ we get an exact description of RMFC where x is the characteristic vector of the vertices to be protected and B is the budget. The first constraint enforces that for each leaf u, one vertex between u and r will be protected, which makes sure that the fire will not reach u. The second constraint ensures that the number of vertices protected after each time step is at most $B \cdot \ell$ and makes sure that we are using no more than B firefighters per time step (see [6] for more details). Note that (as mentioned in [6]), there is an optimal solution to RMFC that protects, with the firefighters available at time step ℓ, only the vertices in V_ℓ. Hence, we can change the above relaxation to one with the same optimal objective value by replacing the constraints $x(V_{\leq \ell}) \leq B \cdot \ell$ by the constraints $x(V_\ell) \leq B$ for all $\ell \in [L]$.

$$\begin{aligned} \min \quad & B & & & (2) \\ & x(P_u) & \geq & 1 \qquad \forall u \in \Gamma \\ & x(V_\ell) & \leq & B \qquad \forall \ell \in [L] \\ & x & \in & \mathbb{R}_{\geq 0}^{V \setminus \{r\}} \end{aligned}$$

Throughout the paper we use a lemma of [1] which basically says that any basic feasible solution of LP(2) (and also LP(1)) is sparse. This is proved for the polytope of the firefighters problem, which has the same LP constraints (just different objective function). Consider any basic feasible solution x to LP(2). One can partition $supp(x) = \{v \in V \setminus \{r\} : x(v) > 0\}$ into

two parts: x-loose vertices and x-tight vertices. A vertex $v \in V \setminus \{r\}$ is x-loose or simply loose if $v \in supp(x)$ and $x(P_v) < 1$. All other vertices in $supp(x)$, which are not loose, will be x-tight or simply tight.

▶ **Lemma 3** (Lemma 6 in [1]). *Let x be a vertex solution to LP(2) for RMFC, then the number of x-loose vertices is at most L, the depth of the tree.*

We will use this property crucially in the design of our algorithm. Also, as noted in [1], we can work with a slightly more general version of the problem in which we have different numbers of budgets/firefighters at each time step: say $B_\ell = m_\ell B$ (for some $m_\ell \in \mathbb{Z}_{\geq 0}$) firefighters for each time step $\ell \in [L]$ while we are still minimizing B. Lemma 3 is valid for this generalization too.

2.2 Height Reduction

The technique of reducing the height of a tree at a small loss in cost (or approximation ratio) has been used in different settings and various problems (e.g. network design problems). For RMFC, Adjiashvili et al. [1] showed how one can reduce an instance of the problem to another instance where the height of the tree is only $O(\log n)$ at a loss of factor 2. In a sense, the tree will be compressed into a tree with only $O(\log n)$ levels. Here we introduce a more delicate version of that compression, which allows for transforming any instance to one on a tree with $O(\frac{\log n}{\epsilon})$ levels at a loss of $1+\epsilon$ in the approximation. Our compression is similar to that of [1] with an initial delay and ratio $1+\epsilon$. One key property we achieve with compression, is that we can later use techniques with running time exponential in the depth of the tree.

Suppose that the initial instance is a tree with L levels and each level ℓ has a budget B_ℓ. To compress the tree to a low height one, we will first do a sequence of what is called up-pushes. Each up-push acts on two levels $\ell_1, \ell_2 \in [L]$ with $\ell_1 < \ell_2$ of the tree, and moves the budget B_{ℓ_2} of level ℓ_2 up to ℓ_1. This means the new budget of level ℓ_1 will be $B_{\ell_1} + B_{\ell_2}$ and for level ℓ_2 it will be 0.

We will show that one can do a sequence of up-pushes such that: (i) the optimal objective value of the new instance is very close to the one of the original instance, and (ii) only $O(\log L/\epsilon)$ levels have non-zero budgets. Finally, 0-budget levels can easily be removed through a simple contraction operation, thus leading to a new instance with only $O(\log L/\epsilon)$ depth. The following theorem is a more powerful version of Theorem 5 in [1] with some improvements such as reducing the loss to only $1+\epsilon$ (instead of 2) and some differences in handling of the first levels.

▶ **Theorem 4.** *Let $G = (V, E)$ be a rooted tree of depth L. Then for some constants $c, d > 0$ (that only depend on ϵ) we can construct efficiently a rooted tree $G' = (V', E')$ with $|V'| \leq |V|$ and depth $L' = O(\frac{\log L}{\epsilon})$, such that:*

(i) If the RMFC problem on G has a solution with budget $B \in \mathbb{Z}_{\geq 0}$ at each level, then the RMFC problem on G' has a solution with non-uniform budgets of $B_\ell = B$ for each level $\ell < c$, and a budget of $B_\ell = m_\ell \cdot B$ for each level $\ell \geq c$, where $m_\ell = \left(\lceil (1+\epsilon)^{(\ell-d+1)} \rceil - \lceil (1+\epsilon)^{(\ell-d)} \rceil\right)$.

(ii) Any solution to the RMFC problem on G', where each level $\ell < c$ has a budget of $B_\ell = B$ and each level $\ell \geq c$ has a budget of $B_\ell = m_\ell \cdot B$ can be transformed efficiently into an RMFC solution for G with budget $\lceil (1+2\epsilon)B \rceil$.

Proof. We start by describing the construction of $G' = (V', E')$ from G. We first change the budget assignment of the instance and then contract all 0-budgets levels.

We set i^* to be the smallest integer such that $(1+\epsilon)^{i^*} \geq \frac{2(1+\epsilon)}{\epsilon^2}$ and we let $c = \lceil (1+\epsilon)^{i^*} \rceil$. The set of levels $\mathcal{L}$ in which the transformed instance will have non-zero budget contains the first $c-1$ levels of G and all the levels $\ell \geq c$ of G such that $\ell = \lceil (1+\epsilon)^i \rceil$ for some $i^* \leq i \leq \frac{\log L}{\log(1+\epsilon)} = O(\log L/\epsilon)$:

$$\mathcal{L} = \left\{ 1 \leq \ell \leq L \quad | \quad \ell < c \quad or \quad \ell = \lceil (1+\epsilon)^i \rceil \text{ for some } i^* \leq i \leq \left\lfloor \frac{\log L}{\log(1+\epsilon)} \right\rfloor \right\}$$

For all other levels $\ell \notin \mathcal{L}$ we first do up-pushes. More precisely, the budget of these levels $\ell \in [L] \setminus \mathcal{L}$ will be assigned to the closest level in $\mathcal{L}$ that is above ℓ (has smaller index than ℓ). We then remove all 0-budget levels by contraction. For each vertex v in a level $\ell_i = \lceil (1+\epsilon)^i \rceil \geq c$ we will remove all vertices in the levels $\ell_i < \ell < \ell_{i+1} = \lceil (1+\epsilon)^{i+1} \rceil$ from its sub-tree and connect all the vertices in level ℓ_{i+1} of its sub-tree to v directly. This leads to a new tree G' with a new set of leaves. Since our goal is to save all the leaves in the original instance, for each vertex $v \in G'$ such that $v \in G$ has some leaves in its contracted sub-tree, we will mark v as a leaf in G' and simply delete all its remaining subtree.

This finishes our construction of $G' = (V', E')$ and it remains to show that both (i) and (ii) hold. Note that the levels in G' correspond to levels of G in $\mathcal{L}$: the first c levels of G' are the same as the first c levels of G; for each $\ell > c$, level ℓ in G' is level $\lceil (1+\epsilon)^{\ell - c + i^*} \rceil$ of G.

Here we want to determine what will be the budget of each level of G'. For each $\ell < c = \lceil (1+\epsilon)^{i^*} \rceil$, the level ℓ of G' is the same as the level ℓ of G and has the same budget $B_\ell = B$, because these levels are not involved in up-pushes. For $\ell = c$, all the budgets from level $\lceil (1+\epsilon)^{i^*} \rceil$ to $\lceil (1+\epsilon)^{i^*+1} \rceil - 1$ in G are up-pushed to this level. This means that the budget for level c in G' is $B_c = \left(\lceil (1+\epsilon)^{i^*+1} \rceil - \lceil (1+\epsilon)^{i^*} \rceil \right) \cdot B$. Now for each $i^* < i \leq \lfloor \frac{\log L}{\log(1+\epsilon)} \rfloor$, all the budgets from levels $\lceil (1+\epsilon)^i \rceil$ to $\lceil (1+\epsilon)^{i+1} \rceil - 1$ in G are up-pushed to level $\lceil (1+\epsilon)^i \rceil$, which becomes level $i - i^* + c$ in G'; this means that the budget for this level of G' will be $\lceil (1+\epsilon)^{i+1} \rceil - \lceil (1+\epsilon)^i \rceil$. Setting $\ell = i - i^* + c$ and $d = c - i^*$, the budget of level ℓ in G', is $B_\ell = (\lceil (1+\epsilon)^{\ell - d + 1} \rceil - \lceil (1+\epsilon)^{\ell - d} \rceil) \cdot B$. To prove (ii), we use the following lemma:

▶ **Lemma 5.** *For any two consecutive levels $\ell \geq c$ and $\ell + 1$ in G', the difference between m_ℓ and $m_{\ell+1}$ is relatively small. More precisely: $m_\ell(1+2\epsilon) \geq m_{\ell+1}$*

Proof. Based on the definition of m_ℓ and $m_{\ell+1}$ we have:

$$\begin{aligned} m_\ell &= \lceil (1+\epsilon)^{(\ell-d+1)} \rceil - \lceil (1+\epsilon)^{(\ell-d)} \rceil \geq (1+\epsilon)^{(\ell-d+1)} - (1+\epsilon)^{(\ell-d)} - 1 \\ \Rightarrow m_\ell(1+\epsilon) &\geq (1+\epsilon)^{(\ell-d+2)} - (1+\epsilon)^{(\ell-d+1)} - (1+\epsilon). \end{aligned} \tag{3}$$

On the other hand:

$$\begin{aligned} m_{\ell+1} &= \lceil (1+\epsilon)^{(\ell-d+2)} \rceil - \lceil (1+\epsilon)^{(\ell-d+1)} \rceil \leq (1+\epsilon)^{(\ell-d+2)} - (1+\epsilon)^{(\ell-d+1)} + 1 \\ &\leq m_\ell(1+\epsilon) + 2 + \epsilon \qquad \text{using (3)} \end{aligned} \tag{4}$$

Also by our choice of c, d and $i^* = c - d$ we can conclude that:

$$\begin{aligned} m_\ell &= \lceil (1+\epsilon)^{(\ell-d+1)} \rceil - \lceil (1+\epsilon)^{(\ell-d)} \rceil \\ &\geq (1+\epsilon)^{(\ell-d+1)} - (1+\epsilon)^{(\ell-d)} - 1 = \epsilon(1+\epsilon)^{(\ell-d)} - 1 \\ &\geq \epsilon(1+\epsilon)^{(c-d)} - 1 = \epsilon \cdot (1+\epsilon)^{i^*} - 1 \geq \epsilon \frac{2(1+\epsilon)}{\epsilon^2} - 1 \\ \Rightarrow \quad m_\ell &\geq \frac{2+\epsilon}{\epsilon} \Rightarrow \quad \epsilon m_\ell \geq 2 + \epsilon. \end{aligned} \tag{5}$$

Combining (4) and (5) completes the proof. ◀

▶ **Corollary 6.** *For each $\ell \geq c$ and each budget $B > 0$:*

$$m_{\ell+1} \cdot B \leq m_\ell \cdot \lceil (1+2\epsilon)B \rceil$$

Notice that in the constructed graph G' for each level $\ell \geq c$, we have $B_\ell = m_\ell \cdot B$. Now consider the instance of the problem on graph G with budget $\lceil (1+2\epsilon)B \rceil$ at each level. We will show that by doing some down-pushes on G (i.e. move the budget of each level to some level down) we can construct G' again where the budget of each level ℓ is $m_\ell \cdot B$, and this means that if G' has a solution with budget $m_\ell \cdot B$ in each level, then G has a solution with uniform budget $\lceil (1+2\epsilon)B \rceil$.

Like before the set of levels $\mathcal{L}$ with non-zero budgets will be the same. Instead of up-pushes, we will down-push the budget from all levels $\ell \notin \mathcal{L}$ to the closest level in $\mathcal{L}$ which is below ℓ (i.e has larger index than ℓ). We will also down-push budget $\lceil 2\epsilon B \rceil$ from each level $\ell < c$ to level $\ell = c$.

By doing the same contraction, for each level $\ell < c$ we will have $B_\ell = B$ and for each level $\ell > c$ we will have $B_\ell = m_{\ell-1} \cdot \lceil (1+2\epsilon)B \rceil$, which is greater than $m_\ell \cdot B$ based on the above lemma.

The only remaining level to consider is level $\ell = c$. For this level, by doing down-pushes, we will have budget $B_c = B + \lceil 2\epsilon B \rceil \cdot c$. Our claim is that this is not less than $m_c \cdot B$, which is equal to $(\lceil (1+\epsilon)c \rceil - c) \cdot B$ (based on the definition of m_c):

$$\begin{aligned} B_c &= B + \lceil 2\epsilon B \rceil \cdot c \\ &\geq B + 2\epsilon B \cdot c = (1 + 2\epsilon c) \cdot B \\ &\geq \lceil 2\epsilon c \rceil \cdot B = \lceil (1+2\epsilon)c - c \rceil \cdot B \\ &\geq (\lceil (1+\epsilon)c \rceil - c) \cdot B = m_c \cdot B. \end{aligned}$$

This will complete the proof of the theorem, because by considering these down-pushes, any solution to the RMFC problem on G', where level $\ell \geq c$ has a budget of $B_\ell = m_\ell \cdot B$ and level $\ell < c$ has a budget of $B_\ell = B$, can be transformed efficiently into an RMFC solution for G with budget $\lceil (1+2\epsilon)B \rceil$. ◀

In the following we assume that the depth of the tree is not more than $\frac{\log n}{\log(1+\epsilon)} + \frac{2(1+\epsilon)}{\epsilon^2}$, so $L = O(\frac{\log n}{\epsilon})$. After finding a solution with budget B for a tree with this height, then we could apply the compression theorem and find a solution for the original tree by having $\lceil \epsilon B \rceil$ more firefighters at each level.

2.3 Overview of the Algorithm

Given an instance $\mathcal{I}$, our first step of the algorithm is to use Theorem 4 to reduce $\mathcal{I}$ to an instance $\mathcal{I}'$ with $L = O(\log n/\epsilon)$ levels. Note that when we use B to refer to *core* budget for instance $\mathcal{I}'$ we mean each level ℓ has budget $m_\ell \cdot B$ for $\ell \geq c$, and budget B for each level $\ell < c$. Also, by $OPT(\mathcal{I}')$ we mean the smallest value B such that $\mathcal{I}'$ has a feasible solution with core budget B as above. By Theorem 4, if we find a solution with *core* budget B for $\mathcal{I}'$ then it can be transformed to a solution for $\mathcal{I}$ with budget $\lceil (1+2\epsilon)B \rceil$. So we focus on the height reduced instance $\mathcal{I}'$ from now on. We present an algorithm such that if $B \geq OPT(\mathcal{I}')$ then it finds a feasible solution to $\mathcal{I}'$ with core budget at most $\lceil (1+\epsilon)B \rceil$. Then, using binary search, we find the smallest value of B_o (for B) for which the algorithm finds a feasible solution. This would give us a solution of budget at most $\lceil (1+\epsilon)OPT(\mathcal{I}') \rceil$, which in turn implies a solution for $\mathcal{I}$ of value at most $\lceil (1+O(\epsilon))OPT(\mathcal{I}) \rceil$.

So let us assume we have guessed a value $OPT(\mathcal{I}') \leq B_o$. We consider LP(2) (with fixed $B = B_o$) for $\mathcal{I}'$ with guessed core budget B_o. Let x^* be a basic feasible solution to this instance. Using Lemma 3 we know that there are at most L loose vertices. As we will see, when B_o is relatively large, i.e. $B_o > \frac{L}{\epsilon}$, then we can easily find an integer solution using core budget at most $\lceil (1+\epsilon)B_o \rceil$ and this yields the desired bound for the original instance.

The difficult case is when B_o is small compared to L. The difficulty lies in deciding which vertices are to be protected by the optimum solution in the top h levels of the tree for some $h = O(\log \log n)$; as if one has this information then we can obtain a good approximation as in [1].

One way to do this would be to guess all the possible subsets of vertices that could be protected by the optimal solution in the first h levels of the tree, but this approach would have a running time far greater than ours. Still, we can solve the problem on instance $\mathcal{I}'$ in quasi-polynomial time using a bottom-up dynamic programming approach. More precisely, starting with the leaves and moving up to the root, we compute for each vertex $u \in V$ the following table. Consider a subset of the available budgets, which can be represented as a vector $q \in [B_1] \times ... \times [B_L]$. For each such vector q and node v, we want to know whether or not using budgets described by q for the subtree T_v (subtree rooted at v) allows for disconnecting v from all the leaves below it, i.e. saving all the leaves in T_v. Since $L = O(\log n/\epsilon)$ and the size of each budget B_ℓ is at most the number of vertices, the table size is $n^{O(\log n/\epsilon)}$). Moreover, it is easy to show that this table can be constructed bottom-up in quasi-polynomial time using an auxiliary table and another dynamic programming, to fill each cell of the table.

This approach would have the total running time of $n^{O(\log n/\epsilon)}$, because of the size of the table. In order to reduce the running time to $n^{O(\log \log n/\epsilon)}$, we would consider each budget vector value rounded up to the nearest power of $(1+\frac{\epsilon^2}{(\log n)^2})$. So, instead of $O(n^L) = n^{O(\log n/\epsilon)}$ many options for budget vectors q, we will have $O((\log n/\epsilon)^{3L}) = n^{O(\log \log n/\epsilon)}$ many options and we will show how by being more careful in our dynamic programming on these budget vectors we can still compute the table in time $n^{O(\log \log n/\epsilon)}$; this leads to an approximation scheme (instead of the exact algorithm) for the instance $\mathcal{I}'$.

3 Asymptotic Approximation Scheme

As mentioned above, first we use the height reduction as discussed in the previous section to reduce the given instance $\mathcal{I}$ to a new one $\mathcal{I}'$ with $L = O(\frac{\log n}{\epsilon})$ levels. We assume we have guessed a value $B_o \geq OPT(\mathcal{I}')$. Recall that, as in the statement of Theorem 4, for some constants c, d (depending on ϵ) the budget of each level $\ell < c$ is $B_\ell = B_o$ and for each level $\ell \geq c$ the budget is $B_\ell = m_\ell \cdot B_o$ where $m_\ell = \left(\lceil (1+\epsilon)^{(\ell-d+1)} \rceil - \lceil (1+\epsilon)^{(\ell-d)} \rceil\right)$.

We consider two cases: (I) when $B_o > \frac{L}{\epsilon}$, and (II) when $B_o \leq \frac{L}{\epsilon}$. For the first case we show how we can find a solution with core budget at most $\lceil (1+\epsilon)B_o \rceil$ by rounding the standard Linear Programming relaxation. For the second case we show how we can use a bottom-up dynamic programming approach to find a quasi-polynomial time approximation scheme.

3.1 Easy Case: $B_o > \frac{L}{\epsilon}$

In this case we consider LP(2) (with fixed $B = B_o$) for this instance. If x^* is a feasible solution to this LP and $B_o > \frac{L}{\epsilon}$ then we add $L \leq \lceil \epsilon B_o \rceil$ extra budget (i.e. number of firefighters) to the first level which is enough to protect all the *loose* vertices. Since by using Lemma 3 we know that there are at most L loose vertices and we can protect them all in the first step using L extra firefighters.

It remains to show that by using a budget of $m_\ell \cdot B_o$ at every level ℓ, for $c \leq \ell \leq L$, and B_o for $\ell < c$, we can protect all the tight vertices and so all the leaves would be saved, by adding only L many extra firefighters to only the first level.

Observe that for each tight vertex v, either $x(v) < 1$, then we would have a loose vertex in P_v, or $x(v) = 1$. In the first case v is already saved by protecting the loose vertices in the first step. If we only consider vertices with $x(v) = 1$, we can see that the solution is integral itself for these vertices. So we have rounded a fractional solution with $B_o > \frac{L}{\epsilon}$ to an integral one by using only $\lceil \epsilon B_o \rceil$ more firefighters just in the first level. In this case we find a feasible solution with core budget $B_o + \lceil \epsilon B_o \rceil$ in polynomial time.

3.2 When $B_o \leq \frac{L}{\epsilon}$

Recall that we have a budget of $B_\ell = B_o < L/\epsilon$ for each level $\ell < c$ and $B_\ell = m_\ell \cdot B_o \leq m_\ell \cdot \frac{L}{\epsilon}$ for each $c \leq \ell \leq L$. We denote by q^* the L-dimensional total budget vector that has $q^*[\ell] = B_\ell$ for each $1 \leq \ell \leq L$. Also for each L-dimensional vector $q \in [B_1] \times [B_2] \times ... \times [B_L]$, we denote by $Q(q)$ the set of all vectors q' such that $q' \leq q$. Suppose that $|Q(q^*)| = m$. We first describe a simpler (and easier to explain) dynamic programming with running time $n^{O(\log n/\epsilon)}$. Then we change it to decrease the running time and have our final approximation scheme with running time $n^{O(\log \log n/\epsilon)}$.

3.2.1 First Algorithm

Our dynamic program (DP) consists of two DP's: an outer (main) DP and an inner DP. In our main DP table A we have an entry for each vertex v and each vector $q \in Q(q^*)$. This entry, denoted by $A[v, q]$, will store whether using budgets described by q for levels of T_v allows for disconnecting v from all leaves below it or not.

More formally, if we assume $v \in V_\ell$, then $A[v, q]$ would be true if and only if there is a strategy for T_v such that (i) all the leaves in T_v are saved, and (ii) the budget for levels of T_v are given by vector q in indices $\ell + 1, \ldots, L$, i.e. $q[\ell + 1]$ for the first level of T_v (direct children of v) , $q[\ell + 2]$ for the second level, and so on.

We compute the entry $A[.,.]$ in a bottom up manner, computing $A[v, q]$ after we have computed the entries for children of v. To compute cell $A[v, q]$, we would use another auxiliary table B. Suppose v has k children $u_1, \ldots, u_k$ and assume that we have already calculated $A[u_j, q']$ for every $1 \leq j \leq k$ and all vectors $q' \in Q(q)$. Then we define a cell in our auxiliary table $B[v, q', j]$ for each $1 \leq j \leq k$ and $q' \in Q(q)$, where $B[v, q', j]$ is supposed to determine if the budget vector q' is enough for the union of subtrees rooted at $u_1, \ldots, u_j$ to save all the leaves in $T_{u_1} \cup \ldots T_{u_j}$ or not, where the total budgets for union of those subtrees are given by q'. We can compute $B[v, q', j]$ having computed $A[u_j, q'']$ and $B[v, q' - q'', j - 1]$ for all $q'' \in Q(q')$. This means that we can compute each cell $A[v, q]$ using auxiliary table B and internal DPs and the running time is $O(n^2 \cdot m^3)$. We need to find $A[r, q^*]$. If this cell is true, then we can save all the leaves of the tree using q^* as the budget vector for each level and if it is false, B_o would not be enough.

The problem is that m_ℓ could be large ($m_L = O(n)$) and so the options we have for the budget of each level is $O(n)$. Recall that we can have $B_o \leq \frac{L}{\epsilon}$ many choices for $q[\ell]$ when $\ell < c$ and $m_\ell \cdot \frac{L}{\epsilon}$ many options when $c \leq \ell \leq L$. Using the definition of the m_ℓ: $m_\ell = O(\epsilon(1 + \epsilon)^{\ell - d})$, and so the total possible different budget vectors we could have is:

$$m = \left(\frac{L}{\epsilon}\right)^{c-1} \times \prod_{\ell=c}^{L} \left(m_\ell \cdot \frac{L}{\epsilon}\right) \quad = \quad \left(\frac{L}{\epsilon}\right)^{c-1} \times L^{L-c+1} \times \prod_{\ell=c}^{L} \left((1+\epsilon)^{\ell-d}\right) \quad = \quad O\left((nL)^L\right)$$

This means that the total running time will be $O(n^L) = n^{O(\log n/\epsilon)}$ and this is an exact algorithm to solve the RMFC problem on instance $\mathcal{I}'$.

3.2.2 Reducing Budget Possibilities

To reduce the running time, we only consider budget vectors where each entry of the vector is a power of $(1+(\epsilon/\log n)^2)$. In this case we have at most $O\big(\log(m_\ell \cdot L)\times(\frac{\log n}{\epsilon})^2\big) = O(\log^3 n)$ many options for ℓth entry of q for each $c \le \ell \le L$, and so $m = O((\log n)^L) = n^{O(\log\log n/\epsilon)}$. Also, we have to show how we can compute the entries of the table in time $n^{O(\log\log n/\epsilon)}$ and why this would give a $(1+\epsilon)$-approximation of the solution. For each real x, let $RU(x)$ denote the value obtained by rounding up x to the nearest power of $(1+(\epsilon/\log n)^2)$. The main idea is that if for each vector q we round up each entry q_i to $RU(q_i)$ and denote the new vector by $RU(q)$ then if $A[v,q] = true$ then $A[v, RU(q)]$ is also true. So we only try to fill in entries of the table that correspond to vectors q where each entry is a power of $(1+(\epsilon/\log n)^2)$. We show this can be done in time $n^{O(\log\log n/\epsilon)}$ and the total loss in approximation is at most $1+\epsilon$ at the root of the tree.

From now on, we assume each vector q has entries that are powers of $(1+(\epsilon/\log n)^2)$; and recall that $Q(q)$ is the set of all such vectors q' such that $q \le q'$ and assume we have already calculated $A[u_j, q']$ for every vector $q' \in Q(q)$ (again with all entries being powers of $(1+(\epsilon/\log n)^2)$).

If we try to compute $A[v,q]$ from $A[u_j,q']$'s the same way, we need to calculate $B[v,q',j]$ for each $1 \le j \le k$ and each time we round up the results of addition/subtractions (such as $q-q'$) to the nearest power of $(1+(\epsilon/\log n)^2)$.

3.2.3 Reducing Height of Inner Table

To compute cell $A[v,q]$ then this round-up operation could happen $k = O(n)$ times and the approximation loss blows up. Instead, we consider a hypothetical full binary tree with root v and leaves (at the lowest level) being $u_1,\dots,u_k$; this tree will have height $O(\log k) = O(\log n)$. Then we define a cell in our auxiliary table for each internal node of this tree. See Figure 1 for an illustration.

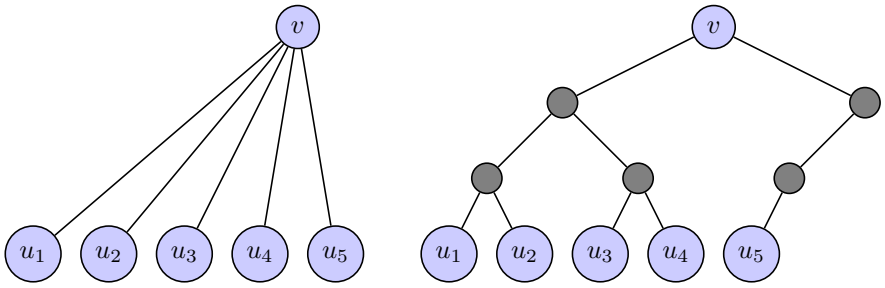

Figure 1 Illustration of the hypothetical full binary tree with root v and leaves $u_1,\dots,u_5$.

More formally we would define a cell in our auxiliary table $B[v,q',j,j']$ for each $0 \le j \le \lceil\log k\rceil$, $1 \le j' \le \lceil\frac{k}{2^j}\rceil$ and $q' \in Q(q)$ with all entries being powers of $(1+(\epsilon/\log n)^2)$, where $B[v,q',j,j']$ is supposed to determine if the budget vector q' is enough for the subtrees rooted at $u_{j_1},\dots,u_{j_2}$, where $j_1 = 2^j\cdot(j'-1)+1$ and $j_2 = min\{2^j\cdot j', k\}$, to save all the leaves in those subtrees, where the total budgets for the union of those subtrees is given by q'.

Similar to what we did before, we can compute $B[v,q',j,j']$ having computed $B[v,q'',j-1,2j'-1]$ and $B[v,RU(q'-q''),j-1,2j']$ (if it exists) for all $q'' \in Q(q')$. At each step we are computing a cell in table B a round-up will be applied to make the result of vector subtraction to be a vector with entries being powers of $(1+(\epsilon/\log n)^2)$. If we can find a q'' such that both $B[v,q'',j-1,2j'-1]$ and $B[v,RU(q'-q''),j-1,2j']$ are true, then $B[v,q',j,j']$ would be true too. Also we can fill $A[v,q]$ by checking the value of $B[v,q_i,\lceil\log k\rceil,1]$.

In the way we construct our auxiliary tables, while computing $A[v,q]$, when v has k children, $\log k$ many round up operations have happened (going up the auxiliary tree with root v) to the solution we found for T_v only in this step. This means that $O(\log k) \leq O(\log n)$ many round-ups could happen to compute entry $A[v,q]$ and the total number of round-ups starting from the values of $A[.,.]$ at a leaf level to $A[r,q]$ (for any q) would be at most $L \times \log n \leq \frac{\log^2 n}{\epsilon}$ and at each round-up we increase our budget by a factor of $(1+(\epsilon/\log n)^2)$. So the total approximation increase while computing the entries for $A[r,.]$ would be at most:

$$\big(1+\frac{\epsilon^2}{(\log n)^2}\big)^{\frac{\log^2 n}{\epsilon}} \quad = \quad 1+O(\epsilon)$$

Observe that for every node v and subtree T_v if there is a solution with budget vectors q then there is a solution with budget vector $RU(q)$ as well. Using this fact we can find a solution with budget vector at most $(1+O(\epsilon))q^*$ if there exists a solution with budget vector q^*. This completes the proof of Theorem 1.

4 Conclusion

In this paper we presented an asymptotic QPTAS for RMFC on trees. More specifically, let $\epsilon > 0$, and $\mathcal{I}$ be an instance of RMFC where the optimum number of firefighters is $OPT(\mathcal{I})$. We presented an algorithm that uses at most $\lceil (1+\epsilon)OPT(\mathcal{I}) \rceil$ many firefighters at each step and runs in time $n^{O(\log\log n/\epsilon)}$. Our result combines a more powerful height reduction lemma than the one in [1] by using dynamic programming to find the solution. We also provide a polynomial time $(5+\epsilon)$-approximation for the problem by applying our height reduction lemma to the algorithm provided in [1] as well as some minor changes to improve the best previously known 12-approximation (Appendix A).

We believe that it should be possible to have an asymptotic PTAS for the RMFC problem. Perhaps one way is to somehow guess the upper part of the optimal solution in polynomial time and then use the LP to round the solution for the height reduced instance for which we initially applied the height reduction lemma.

References

1. David Adjiashvili, Andrea Baggio, and Rico Zenklusen. Firefighting on trees beyond integrality gaps. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2364–2383, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=3039686.3039842`.
2. E. Anshelevich, D. Chakrabarty, A. Hate, and C. Swamy. Approximability of the firefighter problem. In *Algorithmica*, pages 520–536, 2012.
3. L. Cai, E. Verbin, and L. Yang. Firefighting on trees:$(1-\frac{1}{e})$–approximation, fixed parameter tractability and a subexponential algorithm. In *Proceedings of the 19th International Symposium on Algorithms and Computation*, ISAAC '08, pages 258–269. Springer-Verlag, 2008.
4. G. Calinescu, C. Chekuri, M. Pal, and J. Vondrak. Maximizing a monotone submodular function subject to a matroid constraint. In *SIAM Journal on Computing*, pages 1740–1766, 2011.
5. P. Chalermsook and D. Vaz. New integrality gap results for the firefighters problem on trees. In *Approximation and Online Algorithms, Cham, Springer International Publishing*, pages 65–77, 2017.
6. Parinya Chalermsook and Julia Chuzhoy. Resource minimization for fire containment. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1334–1349, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=1873601.1873709`.

7 S. Finbo, A. King, G. MacGillivray, and R. Rizzi. The firefighter problem for graphs of maximum degree three. In *Discrete Mathematics*, pages 2094–2105, 2007.

8 B. Hartnell and Q. Li. Firefighting on trees: how bad is the greedy algorithm? In *Proceedings of Congressus Numerantium*, pages 187–192, 2000.

9 B.L. Hartnell. Firefighter! an application of domination. In *24th Manitoba Conference on Combinatorial Mathematics and Computing*, 1995.

10 Y. Iwaikawa, N. Kamiyama, and T. Matsui. Improved approximation algorithms for firefighter problem on trees. In *IEICE Transactions on Information and Systems*, pages 196–199, 2011.

11 A. King and G. MacGillivray. The firefighter problem for cubic graphs. In *Discrete Mathematics*, pages 614–621, 2010.

12 Euiwoong Lee. Improved hardness for cut, interdiction, and firefighter problems. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 92:1–92:14, 2017. URL: `https://doi.org/10.4230/LIPIcs.ICALP.2017.92`.

13 J. Vondrak. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 67–74, 2008.

A Polynomial $(5+\epsilon)$-Approximation for RMFC

In this section we show how the approach introduced in [1] can be adapted so that along with our height reduction lemma gives a $(5+\epsilon)$-approximation. We largely follow the proof of [1] only pointing out the main steps that need slight adjustments. We assume the reader is familiar with that proof and terminology used there.

Let x be a fractional solution to LP(2). We define W_x as the set of leaves that are (fractionally) cut off from r largely on low levels, i.e. there is high x-value on P_u on vertices far away from the root. We first start by recalling Theorem 12 from [1] which basically says that we can round an LP solution to an integral one by increasing the core budget B by a small constant such that W_x can be saved.

▶ **Theorem 7** (modified version of Theorem 12 in [1]). *Let $B \in \mathbb{R}_{\geq 1}$, $\mu \in (0,1]$, and $h = \lfloor \log_{1+\epsilon} L \rfloor$. Let $x \in$ LP(2) with value B and $supp(x) \subseteq V_{>h}$, and we define $W = \{u \in \Gamma | x(P_u) \geq \mu\}$. Then one can efficiently compute a set $R \subseteq V_{>h}$ such that:*

- *$R \cap P_u \neq \emptyset \quad \forall u \in W$, and*
- *There is an integral solution $z = y_1 + y_2$ to LP(2), which is a combination of two integral solutions y_1 and y_2 with value $B' = \frac{1}{\mu}B$ and 1 respectively such that $supp(y_1) \subseteq V_{>h}$ and $supp(y_2) \subseteq V_{\leq h}$.*

Proof. The proof would be very similar to the proof of Theorem 12 in [1], and the only difference is in providing the extra budget for protectecting the loose vertices in $V_{>h}$. They changed B to $B+1$ at level $h+1$ to provide this required budget. It that was enough, because the budget in the reduced instance is $B_{h+1} = 2^{h+1} \cdot B$ at this level, and so by this change $2^h = L$ many more firefighters are available and they are enough to protect all the loose vertices. But we need to change B to $B+1$ on all levels 1 to h, to have L many more firefighters for protecting all the loose vertices. This is because our budget in the reduced instance is $B_\ell = B$ when $1 \leq \ell < c$ and $B_\ell = m_\ell \cdot B$ when $c \leq \ell \leq L$. So by this change, we should have $c-1$ more firefighters in total for the first $c-1$ levels and $\sum_{\ell=c}^{h} m_\ell$ many more firefighters for levels c to h and the total would be $(1+\epsilon)^h = L$, which is enough to protect all the loose vertices. But the difference in our integral solution is that all the added budgets are from levels 1 to h (one for each level), and the remaining integral solution, which is $\frac{1}{\mu}$ feasible, is the subset of $V_{>h}$. This completes the proof of this theorem. ◀

Similar to [1], we consider two cases based on how B compares to $\log L$. When $B \geq \log L$, we will have a 3-approximation for the reduced instance, by first solving the LP(2). This is similar to Theorem 13 in [1] and consistent with our height reduction lemma:

▶ **Theorem 8** (modified version of Theorem 13 in [1]). *There is an efficient algorithm that computes a feasible solution to a compressed instance of RMFC with budget at most* $3B_{OPT}$ *when* $B \geq \log L$.

Proof. Assume x is a fractional LP(2) solution with value B. Then we use Theorem 7 and set $\mu = 1/2$ to obtain an integral solution z, which saves $W = \{u \in \Gamma | x(P_u) \geq \mu\}$, by core budget 1 at each level $1 \leq \ell \leq h$ and $2B$ at each level $h+1 \leq \ell \leq L$. Note that we can now transfer the 1 unit of budget from the very first level $\ell = 1$ to level $h+1$ and change the core budget $2B$ to $2B+1$ on this level and remove that extra budget from the very first level. This is because these extra firefighters from levels 1 to h are supposed to protect the loose vertices, which are in $V_{>h}$. By doing so we have an integral solution z such that the core budget is 0 in the first level, 1 in levels 2 to h, $2B+1$ at level $h+1$, and $2B$ at level $h+2$ to L. Now consider leaves $\Gamma \setminus W$. If we write another LP similar to LP(2), but specifically to save only these leaves by only protecting the vertices in $V_{\leq h}$, this LP would be feasible. Because all these vertices had $x(P_u) \cap V_{\leq h} \geq 0.5$, and so, $2x$ restricted to the vertices in $V_{\leq h}$, would be a feasible solution to this LP. Hence, we can find the optimal solution to this LP call it y. Based on Lemma 3, there would be at most $h = \log L$ many loose vertices all in $V_{\leq h}$, and so by adding $B > \log L = h$ many firefighters in the first level we would be able to protect all these y-loose vertices. Then all other remaining vertices could be saved by core budget $2B$. Putting these two solutions together (for saving W and $\Gamma \setminus W$) we have found an integral solution to save all the leaves, by having core budget $3B$ in the first level, $2B+1$ in levels 2 to $h+1$, and $2B$ at the remaining levels. This completes the proof of this theorem. ◀

We use the same terminology defined before Lemma 14 in [1], in particular for clean set pairs of vertices A, D. Suppose (A, D) is a clean pair compatible with OPT, i.e. $A \cup D \subseteq V_{\leq h}$, $A \subseteq OPT$ and $D \cap OPT = \emptyset$, for $h = \log \log L$ and $LP(A, D)$ by adding two sets of constraints to LP(2) to force the solution to pick all vertices in A and not picking all vertices in D as well as the vertices in their path to the root. Also for each fractional solution to this LP let $W_x = \left\{u \in \Gamma \middle| x(P_u \cap V_{>h}) \geq \frac{1}{1+\epsilon}\right\}$ to be the set of leaves cut off from the root by an x-load of at least $\mu = \frac{1}{1+\epsilon}$ within bottom levels (we changed $\frac{2}{3}$ to $1/(1+\epsilon)$ from [1]). For each $u \in \Gamma \setminus W_x$, let $f_u \in V_{\leq h}$ be the vertex closest to the root among all vertices in $(P_u \cap V_{\leq h}) \setminus D$, then define $F_x = \{f_u | u \in \Gamma \setminus W_x\} \setminus A$. It follows that no two vertices of F_x lie on the same leaf-root path. Furthermore, every leaf $u \in \Gamma \setminus W_x$ is part of the subtree T_f for precisely one $f \in F_x$. Also lets define $Q_x = V_{\leq h} \cap (\cup_{f \in F_x} T_f)$.

Now we are ready to provide our modification of Lemma 14 in [1] when $B < \log L$:

▶ **Lemma 9** (modified version of Lemma 14 in [1]). *Let* (A, D) *be a clean pair of vertices* (A, D)*, which is compatible with* OPT*, and let* x *and* y *be optimal solutions to* $LP(A, D)$ *and* $LP(A, V_{\leq h} \setminus A)$ *with objective function* B *and* $\hat{B}$ *respectively. Then, if* $OPT \cap Q_x = \emptyset$*, we have* $\hat{B} \leq (2+\epsilon)B_{OPT}$.

Proof. The proof is similar to the proof of Lemma 14 in [1] and the first difference is that we changed $\frac{2}{3}$ to $\frac{1}{1+\epsilon}$ in the definition of W_x. First of all we can have a fractional solution that saves W_x with picking only vertices from $V_{>h}$. This is because $(1+\epsilon)x$ restricted to levels $h+1$ to L would save W_x. Now partition $\Gamma \setminus W_x$ into two groups. The leaves that OPT cut them from the root by protecting a vertex in $V_{\leq h}$, denote them by W_1, and W_2

are the leaves that OPT is cutting them in levels $h+1$ to L. By finding such (A, D), we are actually saving W_1. and for W_2 there is an integral solution with core budget B_{OPT}, which is restricted to levels $h+1$ to L. So the optimum solution to $LP(A, V_{\leq h} \setminus A)$ would not use more than $(1+\epsilon)B_{OPT} + B_{OPT}$ as the core budget in levels $h+1$ to L. This completes the first part of lemma. To round this fractional solution to an integral one which saves W_x and W_2 (note that W_1 is saved already by the choice of A and D), we use the same technique as Theorem 7.

We need to first find an integral solution restricted to levels $h_1 = \log L$ to L that saves the leaves with $y(P_u \cap V_{>h_1}) \geq \frac{1}{2(1+\epsilon)}$ by adding one core budget to levels 1 to h_1 and then write another LP restricted to levels h to h_1. Then we find another integral solution restricted to levels h to h_1 by adding another core budget to levels 1 to h that saves all the remaining leaves, which for sure has $y(P_u \cap V_{>h} \cap V_{\leq h_1}) \geq \frac{1}{2(1+\epsilon)}$. Finally we would have an integral solution with core budget $B_{OPT} + 2$ for the first h levels, $2(2+\epsilon)B_{OPT} + 1$ for levels $h+1$ to h_1 and $2(2+\epsilon)B_{OPT}$ for levels h_1 to L. This completes the proof of this lemma. ◀

The only remaining thing is to show how we can find such (A, D) pair of vertices in polynomial time that follows in the exact same way of Lemma 15 in [1]. and the only difference is the running time, which is still polynomial. In their proof they have used the fact that for each leaf $u \in \Gamma \setminus W_x$, we have $x(P_u \cap V< h) > \frac{1}{3}$, and here we can say $x(P_u \cap V< h) > 1 - \frac{1}{1+\epsilon} > \epsilon$ that would only change the constant factor in the actual running time of $O(\log L)^{O(\log L)}$. So the total running time would be still polynomial. This means that we are able to find a $(5+\epsilon)$-approximation for the reduced instance of the RMFC problem, and then it leads to the $(5+\epsilon)$-approximation for the RMFC problem.